Jon Postel

July 81

A Virtual Terminal Management Model

RFC 782

prepared for

Defense Communications Agency
WWMCCS ADP Directorate
Command and Control Technical Center
11440 Isaac Newton Square
Reston, Virginia 22090

by
Jose Nabielsky
Anita P. Skelton

The MITRE Corporation
MITRE C(3) Division
Washington C(3) Operations
1820 Dolley Madison Boulevard

TABLE OF CONTENTS

		Page
LIST OF ILLUSTRATIONS		vi
1.0	INTRODUCTION	1
1.1	The Workstation Environment	1
1.2	Virtual Terminal Management	2
1.3	The Scope	3
1.4	Related Work	4
2.0	THE VTM MODEL	5
2.1	The VTM Model Components	7
2.2	The Virtual Terminal Model	10
	2.2.1 Virtual Terminal Connectivity	11
	2.2.2 Virtual Terminal Organization	11
	2.2.2.1 The Virtual Keys	12
	2.2.2.2 The Virtual Controller	12
	2.2.2.3 The Virtual Display	12
	2.2.3 Virtual Terminal Architecture	13
	2.2.3.1 Communication Variables	13
	2.2.3.2 Virtual Display with File Extension	13
	2.2.3.3 Virtual Display Windows	14
2.3	The Workstation Model	17
	2.3.1 The Adaptation Unit	17
	2.3.2 The Executive	18
REFERENCES		19

LIST OF ILLUSTRATIONS

Figure Number		Page
2.1	The Virtual Terminal Model	7
2.2	The Workstation Model	8
2.3	VT 0 (expanded from previous figure)	9
2.4	The Domains	14

1.0 INTRODUCTION

Recent advances in micro-electronics have brought us to the age of the inexpensive, yet powerful, microprocessor. Closely resembling the advances of the 1960's which brought about the transition from batch processing to time-sharing, this technological trend suggests the birth of decentralized architectures where the processing power is shifted closer to the user in the form of intelligent personal workstations. The virtual terminal model described in this document caters to this anticipated personal computing environment.

1.1 The Workstation Environment

A personal workstation is a computing engine which consists of hardware and software dedicated to serve a single user. As part of its architecture, the workstation can invoke the resources of other, physically separate components, effectively extending this personal environment well beyond the bounds of the single workstation.

In this personal environment, processing resources previously shared among multiple users now become dedicated to a single one, with a large part of these resources summoned to provide an effective human-machine interface. As a consequence, modalities of input and output that were unfeasible under the time-shared regime now become a part of a conversational language between user and workstation. Due to the availability of processing cycles, and the closeness of the user devices to these cycles, the workstation can support interactive devices, and dialogue modes using these devices, which could not be afforded before.

The workstation can provide the user with the mechanisms to conduct several concurrent conversations with user-agents located elsewhere in the global architecture. One such mechanism is the partitioning of the workstation physical display into multiple logical displays, with one or more of these logical displays providing a dedicated workspace between user and agent.

The nature of the conversations on these logical displays need not be limited to conventional alphanumeric input and output. Conversations using input tools such as positioning and pointing devices (e.g., mouse, tablet, and such), and using high-resolution graphics objects for output (e.g., line drawings, raster blocks and images, possibly intermixed with text) should be possible on one or more of these screens.

Moreover, as long as the technological trend continues in its predicted path, one can postulate a workstation which could support by the mid 1980's multi-media conversations using voice and video,

synchronized with text and graphics. At present, multi-media information management (i.e., acquisition, processing, and dissemination) is an active research area, but eventually it will become an engineering problem which, when solved, will add a new dimension to already feasible modes of interaction between user and workstation.

1.2 Virtual Terminal Management

All virtual terminal protocols (VTPs) provide a vehicle for device-independent, bi-directional, 8-bit byte oriented communications between two VTP users. Most Vo so by invoking a device abstraction of real terminals, called a virtual terminal.

As with a real device, a virtual terminal has a well-defined architecture with its own character sets and functions. A VTP uses the architectural features of the virtual terminal to provide a common language, an intermediate representation, between its two communicating entities. However a VTP user does not communicate directly with this virtual terminal. A function of a VTP is the local mapping between the site-specific order codes and the virtual terminal domain, thus allowing this adaptation to be transparent to the VTP users.

The model of a personal workstation as a dedicated device with considerable resources affects the way we conceptualize the architecture of virtual terminals, both in breadth and depth of function. It also affects the way we view the virtual terminal vis-a-vis its local correspondents, the personal workstations, and its remote correspondents, the other virtual terminals.

This document presents a radical view of virtual terminals as resource sharing devices. The classical concept of a virtual terminal as a two-way device with a limited architecture has been dismissed. Instead, we view a virtual terminal as an n-way device with multiple correspondents sharing access to its virtual "keyboard" and "display." In this model, a virtual terminal has two kinds of correspondents: adaptation units, and other virtual terminals. The adaptation units serve as interface agents between the virtual terminal and its users, providing the step transformation between the user-specific order codes and the virtual terminal interface language. In turn, the other virtual terminals are cooperating co-equals of the virtual terminal, interacting with it to maintain global control and data store synchrony. Resembling the administrator of a local copy of a distributed data base, the virtual terminal interacts with the other virtual terminals (the remote data base managers) and with the local adaptation units (the data base transformers) to provide read, write, and modify access to its local

data store (the local copy of the distributed data base), while providing concurrency control to maintain a "single user view" when so desired.

To communicate with its correspondents, a virtual terminal uses two virtual languages. In the case where the correspondent is another virtual terminal, it uses the language of the virtual terminal protocol; in the case where the correspondent is an adaptation unit, it uses an interface language closer to the physical architecture of the end-user, but a virtual language nevertheless.

In essence, the virtual terminal has become a device in its own right, free from a single physical realization and also dedicated ownership. As a result, a single workstation not only may request any number of virtual terminals, but a number of workstations may share -- and interact with -- a particular virtual terminal.

The functional breadth of virtual terminals has been augmented by the concept of virtual terminal classes. Each class is an abstraction of a particular device architecture. There are stream, line, logical page, physical page, and graphics virtual terminals, all made up of: a class-constrained data structure and its attendant operations (the virtual display); a general controlling element (the virtual controller); and an input selector (the virtual keys).

Finally, the functional depth of the virtual terminal has been extended by architectural features previously unavailable. The virtual terminal becomes a multi-user device with a non-volatile virtual display available for selective viewing. These concepts are discussed is some detail in the chapter that follows.

1.3 The Scope

An overview of the virtual terminal model and the management of communicating virtual terminals is presented. A detailed design description of the data structures and accompanying addressing functions has been completed. The operations and control mechanisms are less complete. Before the design is solidified, an initial mimimal implementation will be made to validate the model.

This document represents work in progress; current international interest in virtual terminal protocols has motivated us to submit this as an example of mechanisms that a virtual terminal should support. The model provides a framework for supporting device and processing capabilities not yet commonly available. A virtual terminal protocol standardization effort may not want to include all the mechanisms that are described here, but it is our contention that one should not preclude these extensions for the future.

1.4 Related Work

The concepts presented in this document are the offspring of previous work in the area of personal computing, and of user interfaces to (distributed) systems. The bibliography at the end of the document collects this material. In particular, we want to acknowledge the work done at the University of Rochester on virtual terminals,(6) work which has influenced to a large degree how we view user interfaces through a display.

2.0 THE VTM MODEL

This section describes a virtual terminal management (VTM) model whose architecture not only derives from a quest for device-independent, terminal-oriented communications, but more importantly from a desire to provide effective human-machine interfaces.

The VTM architecture is a multi-user structure which spans several building blocks. The underlying foundation to this structure is provided by the cooperating virtual terminals. Under the VTM model, these cooperating virtual terminals are viewed as device abstractions, all with a common architecture, exchanging virtual terminal protocol items to update each other's view of the world. Resting on this foundation lie the adaptation units. Associated with a single end-user, an adaptation unit provides the step transformation between user and virtual domains. In a sense the adaptation unit is also a virtual terminal, although one which is much closer to the architecture of the end-user. Finally, on top of this supporting structure are the end-users, the application and human processes, all interacting towards a common goal.

Before embarking on a description of the VTM model components, we present the set of capabilities the VTM model provides its end-users, either human or application. After all, the motivation for the model and its underlying concepts stems from our desire to provide productive user environments.

HUMAN <---> WORKSTATION

- Multiplexing the workstation physical display both in time and space.

  The workstation assigns to each user conversation a logical terminal with a well-distinguished logical display. Under the user control, the workstation maps these logical displays on non-overlapping areas of the physical display, providing a dedicated workspace between user and correspondents. Limited only by the area of the display, many logical displays could be mapped at one time, each providing display updates when so required. Since the area of the display is a scarce resource, not all logical displays need be mapped at the same time. Therefore, the workstation may roll-out and roll-in selected displays under the user control, thereby also multiplexing the physical display in time.

- Multiplexing the workstation input devices in time.

The input devices always map to a single user conversation (i.e., a single logical terminal). However, the user can select a new logical terminal by some well-defined interaction (e.g., depressing a function key, using a pointing device, and such), effectively switching the ownership of the input tools.

- Concurrent multi-mode use of the workstation.

  The capabilities of the workstation limit the scope and character of the individual conversations. If the workstation supports rubout processing (i.e., erase operations on lines and characters), then the logical terminals can be independent, scrolling "terminals," some page-oriented, others line-oriented. If the architecture of the workstation supports graphics objects as primitive objects then so can the individual logical terminals. As a consequence, while some logical terminal displays may be dedicated to alphanumeric output, others may include raster graphics and imaging data together with positioned text.

- The sharing of a single logical terminal among several users.

  Several end-users may link to a single logical terminal. All linked parties are viewed by the shared "device" as both input sources and output sinks. As a consequence this device sharing need not be limited only to the sharing of device output. In general, each linked party may have full read and write access to the logical terminal, if it so desires.

- Selective viewing on a logical terminal display.

  In the user's view, a logical terminal display is a user-specified window on a potentially larger structure, the "device" display. This window provides the "peephole" through which the device display is viewed. The portion of the device display mapped on this window is not limited to its "present contents." Under the user control, the workstation may invoke the viewing of past activity on a logical terminal display when the device display is I/O file-extended. Since the window mechanism is an integral part of the device architecture, it is available on all logical terminal displays. Furthermore, the viewing of past activity does not affect others sharing access to the device.

- Discarding, suspending, and resuming the output of a logical terminal always under user control.

  As part of the user interface, the workstation provides simple "keys" through which the user controls the output on a logical terminal display. These workstation "keys" need not be physical keys, but could be other input tools used for this purpose (e.g., analog dials, hit-sensitive areas on the physical display, and such). In any event, through the auspices of the workstation, the user's control requests translate into the proper commands to the "device" associated with the logical terminal.

APPLICATION <---> ADAPTATION UNIT

- A logical view of real devices.

  For each real terminal architecture, one canonical representation: a logical device.

- For a particular logical device, several possible interaction paradigms.

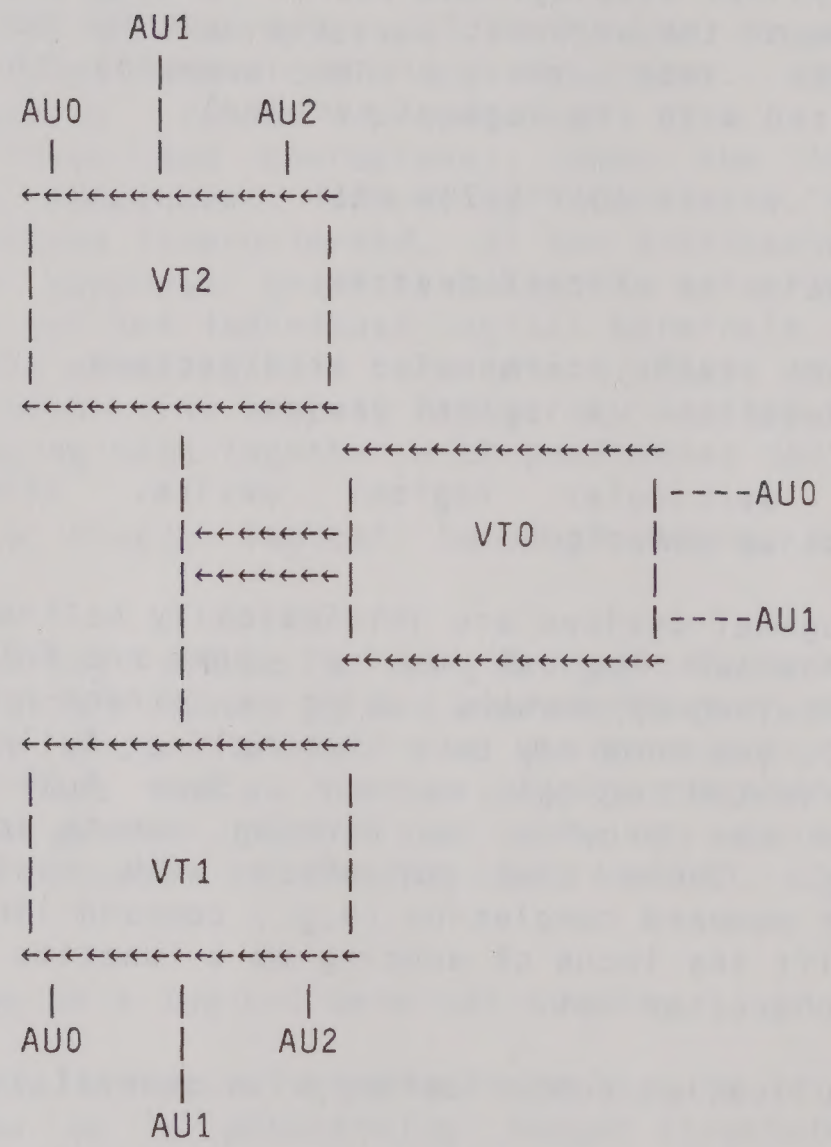
  Some logical devices are intrinsically half-duplex (e.g., a page-oriented logical device), some are full-duplex (e.g., communicating processes using a stream-oriented logical device), and some may be either half or full-duplex (e.g., a line-oriented logical device). Some full-duplex logical devices can provide no echoing, remote echoing, or local echoing. Those that interface with applications that support command completion (e.g., command-line interpreters) can shift the locus of echoing as a function of a dynamic break character set.

- One application communicating with several logical devices.

  As part of an application's model of interaction, an application may "own" several logical devices. For example, an editor could use a line-oriented logical device to gather top-level commands, and a page-oriented logical device to provide editing workspace.

2.1 The VTM Model Components

The virtual terminal management model consists of two major components: the virtual terminal model, and the workstation model (see Figures 2.1, 2.2, and 2.3 respectively).



VT = VIRTUAL TERMINAL
AU = ADAPTATION UNIT

FIGURE 2.1 - THE VIRTUAL TERMINAL MODEL

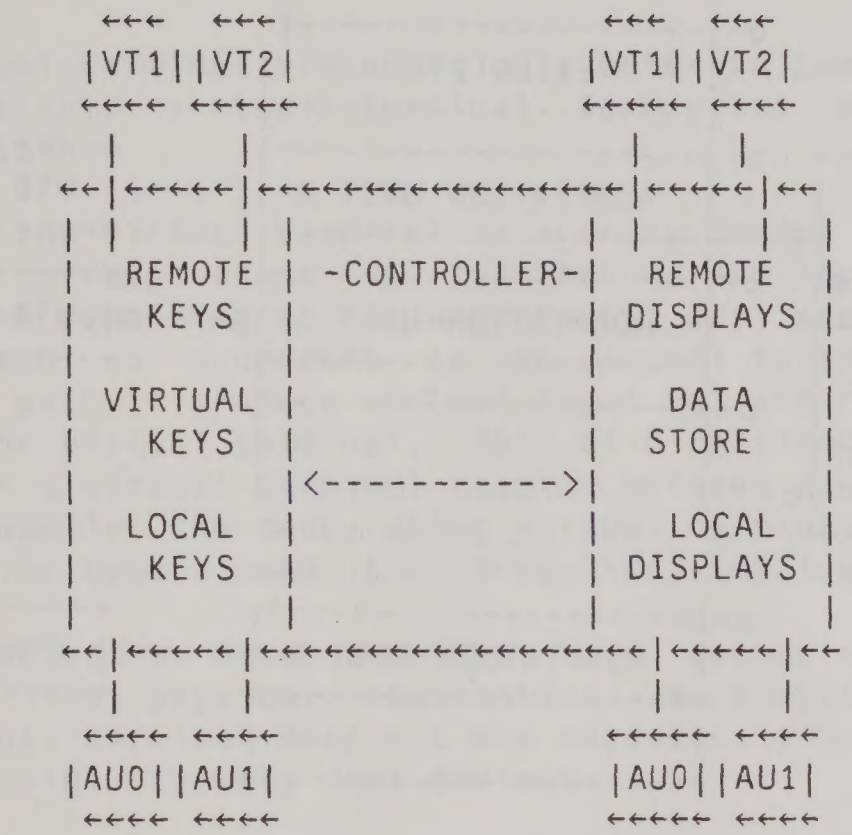


FIGURE 2.2 -- VT0 (expanded from previous figure)

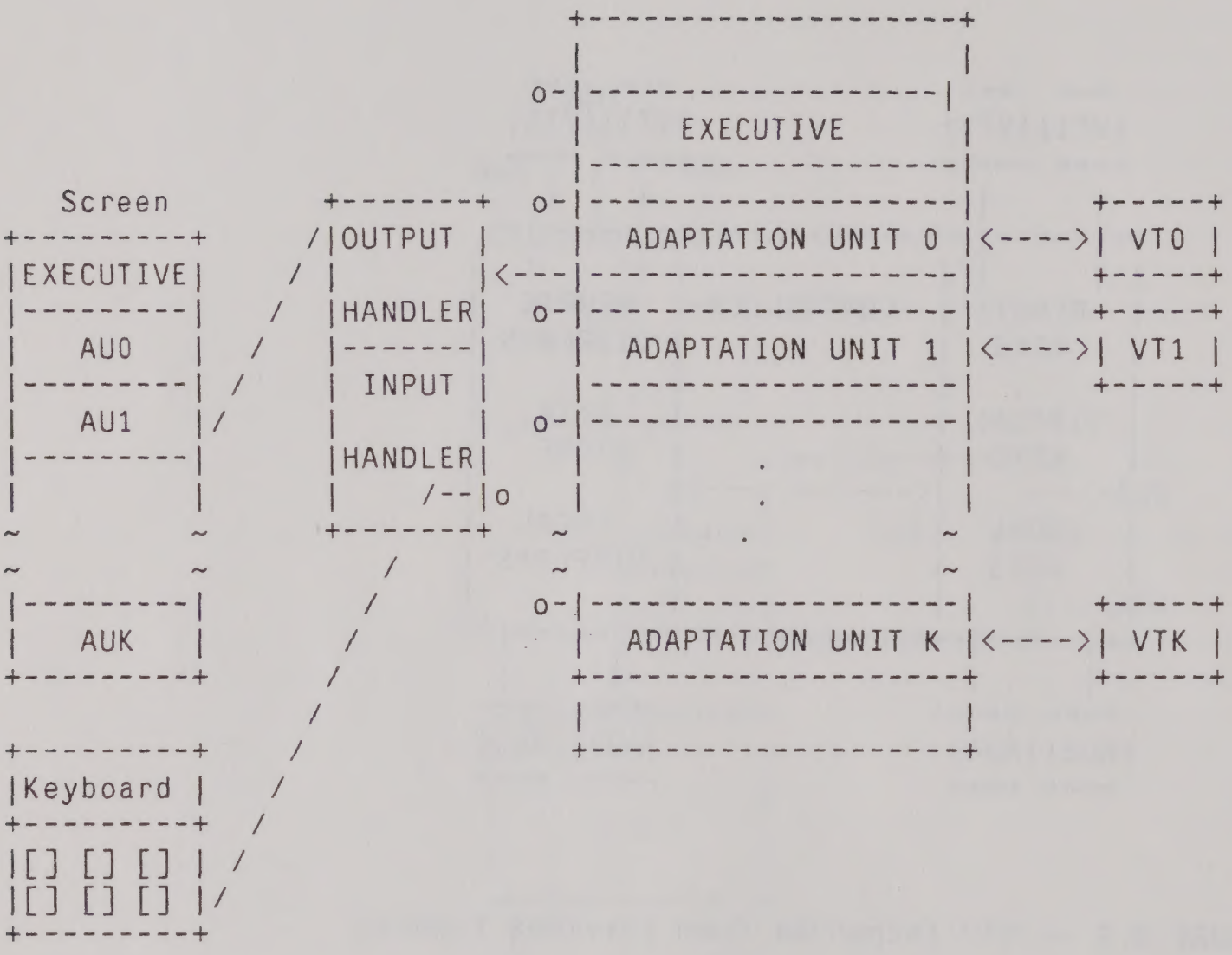


FIGURE 2.3 - THE WORKSTATION MODEL

The first component embodies the canonical device, while the second component includes the adaptation unit and its associated environment. Each component will be described in turn below.

2.2 The Virtual Terminal Model

The objective of virtual terminal protocols is to provide the users of the service with a common, logical view of terminals. The common user view is attained through a standard, protocol-wide representation of a canonical terminal, the virtual terminal. This

permits the exchanges between users of the protocol to be free of device-specific encodings.

The design postulates an integrated virtual terminal model which extends the nature and scope of this canonical device in several important ways. The major aspects of the model, its connectivity, its organization, and its architecture are described below.

2.2.1 Virtual Terminal Connectivity

Most virtual terminal protocols only cater to two-way dialogues in which a single virtual terminal terminates each end of the communication path.

We define the virtual terminal as a n-way device where one or more of the correspondents to this device are local users of the service, and the remaining correspondents (if any) are peer virtual terminals. Each correspondent to the virtual terminal has its own bi-directional path to produce virtual input to, and receive virtual output from, the virtual terminal. This bi-directional path provides the vehicle for a virtual terminal session between user and virtual terminal. Globally, the cooperating virtual terminals and these bi-directional paths span a dendritic (tree-like) topology.

It is important to note that we have decoupled the virtual terminal from its physical realization, a single real terminal. Indeed, a virtual terminal does not map necessarily to just one real device, but possibly to many real devices.

The virtual terminal is viewed ultimately as a well-defined data structure which provides its correspondents with a non-dedicated virtual terminal service. And these correspondents may have read only, write only, or read/write access rights to this data structure.

2.2.2 Virtual Terminal Organization

The virtual terminal is an abstraction; its organization, the building blocks which make up the virtual terminal, is the result of a feature extraction of the real terminal that it is tailored to support.

We have conceptualized the virtual terminal as a meta-terminal (i.e., the terminal of terminals). The meta-terminal is composed of three well-distinguished building blocks: virtual keys, a virtual controller, and a virtual display.

2.2.2.1 The Virtual Keys. The analog of the virtual keys is the physical keyboard of real terminals. However, while the keys of a physical terminal are controlled by a single manual process, these virtual keys can be activated by multiple, concurrent entities (the virtual terminal correspondents). Each correspondent of the virtual terminal, be it a user of the service or a peer virtual terminal, has its input stream to the meta-terminal terminated at the virtual keys. The virtual keys provide the control of access of input streams to the meta-terminal.

2.2.2.2 The Virtual Controller. The virtual controller provides virtual terminal session management. It manages the establishment and termination of a virtual terminal session with a correspondent; supports the possible negotiation and renegotiation of the session attributes; and enables the deactivation and later activation of the session. The virtual controller also provides virtual terminal signalling control by managing the out-of-band signals addressed to the virtual terminal.

2.2.2.3 The Virtual Display. The virtual display is the dynamic component in the meta-terminal organization. For each class of real device (e.g. stream, line, page, or graphics-oriented devices) there is a corresponding virtual terminal class. The organization of the virtual terminal data structure is class-specific. A virtual terminal models a particular terminal class when it is 'fitted' with the proper data structure manager or virtual display. This binding need not be static (e.g., a line-class specialist, and so forth), but could be result of decisions made at "run-time" by applying the principle of negotiated options.

The virtual display manages the data structure associated with the meta-terminal and performs operations on the control and data elements of the structure. As a direct consequence of these operations on the meta-terminal data structure, the virtual display may generate display updates to one, some, or all of the correspondents. All virtual terminal output streams originate at the virtual display.

Different virtual terminal classes are spawned by different "kinds" of virtual displays, and this is realized in one of two ways. For character-oriented virtual devices, it is possible to use a single, wide-scoped virtual display with a character-oriented data structure by constraining it to conform to the model of the device class (e.g., line-oriented devices must be constrained to line-access rules). For non character-oriented virtual devices (e.g., graphics devices), an altogether different virtual display must be used with

properties better suited for the new domain (e.g., a graphics virtual display based on a structured display file).

2.2.3 Virtual Terminal Architecture

The commands, and associated parameters, which are available to the users of the virtual terminal constitute the virtual terminal architecture. The commands available to a user -- to request the virtual controller to establish, abort, or close a session, and discard, suspend, or resume output -- remain invariant to the virtual terminal class. However, as one would expect, the user interface to the virtual display depends on the nature of this data structure.

Three important architectural features of the meta-terminal are: the concept of communication variables, the notion of a file-extended virtual display, and the concept of virtual display windows. Each of these concepts are a part of the meta-terminal architecture because they are apparent to the users of the virtual terminal.

2.2.3.1 Communication Variables. Each component of the meta-terminal (i.e., virtual keys, controller, display) is assigned a standard, protocol-wide name which we call a communication variable. The communication variable is a part of the header of each command to the virtual terminal (i.e. protocol item). It permits better management of the virtual terminal command name space, and also provides the virtual keys with an easy mechanism to select the destination of the request. It must be noted that nothing in the model precludes the addition of more virtual entities to the meta-terminal, such as auxiliary virtual devices and signalling devices. The use of communication variables provides a naming hierarchy which alleviates the problems of device selection and command name allocation in the case of such extensions.

2.2.3.2 Virtual Display with File Extension. The virtual display is the immediate manager of the meta-terminal data structure. When the virtual display is provided with an I/O file extension, it is possible to introduce the concept of a stable-store data structure, a data structure whose contents are stored in backing store (e.g., disk). If the virtual display is provided with this file extension capability (a local option with no end-to-end significance), then the meta-terminal data structure inherits the spatial and temporal attributes (dimensions and time-to-live) of the associated file. Such a virtual display, coupled with the concept of virtual display windows below, provides the users of the service with a very powerful tool.

2.2.3.3 Virtual Display Windows. To communicate with a virtual terminal, each real device uses an adaptation unit as its interface entity (this adaptation unit is a part of the workstation model, see section 2.3). What is important to note is that the adaptation unit provides the transition between the device-specific domain, the device workspace, and the virtual domain, the master workspace (see Figure 2.4).

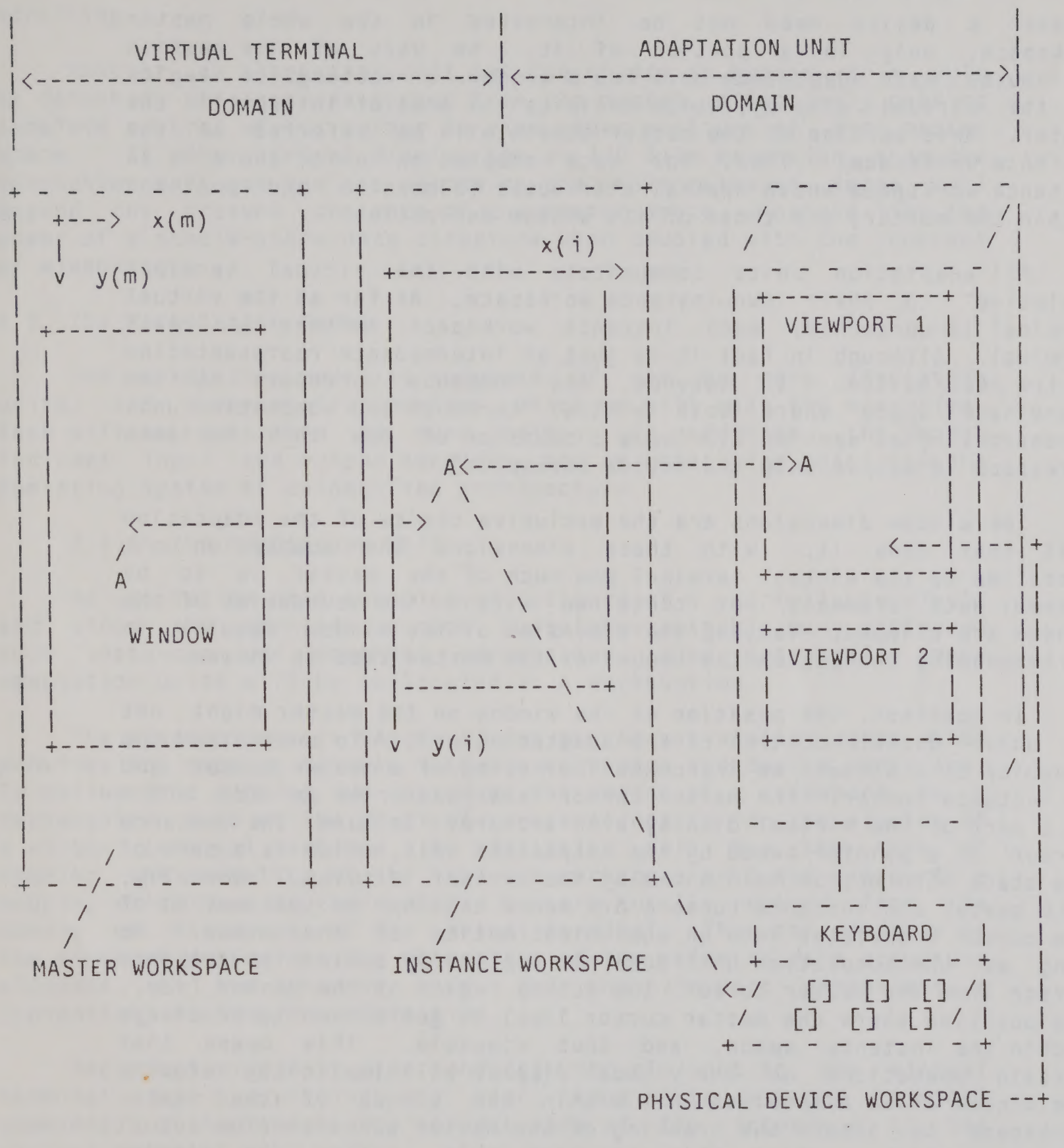


FIGURE 2.4 -- THE DOMAINS

However a device need not be interested in the whole master workspace, only in a portion of it. As part of its session attributes, each adaptation unit has a window, a rectangular region in the virtual display, which delimits its area of interest in the master. This portion of the master domain will be referred as the instance workspace. Then, for each adaptation unit, there is an instance workspace whose spatial attributes (dimension and position within the master) are those of its window definition.

All adaptation units communicate with the virtual terminal "relative" to their own instance workspace. As far as the virtual terminal is concerned, each instance workspace defines a "real" terminal, although in fact it is just an intermediate representation of the real device. In essence, the instance workspace is the coordinate space where both virtual terminal and adaptation unit rendezvous. (See section 2.3 for a discussion of how this instance workspace is mapped onto the device workspace).

The window dimensions are the exclusive choice of the adaptation unit that owns it. With these dimensions the adaptation unit specifies to the virtual terminal how much of the master is to be viewed; data elements not contained within the boundaries of the window are clipped. Varying the dimension of the window results in corresponding changes on the amount of the master that is viewed.

In contrast, the position of the window on the master might not be under direct control of the adaptation unit. To understand the dynamics of a window, we introduce the notion of a master cursor and an instance cursor. The master cursor is a read/write pointer, which is a part of the virtual display architecture. In turn, the instance cursor is a pointer owned by the adaptation unit, which is a part of the state information maintained by the virtual display. Normally, both master and instance cursors are bound together so that motion of one cursor translates into an equivalent motion of the other. As long as the adaptation unit does not explicitly unbind its instance cursor from the master cursor, the active region of the master (i.e., the position where the master cursor lies) is guaranteed to be always within the instance space, and thus viewable. This means that certain operations on the virtual display will implicitly relocate the window of an adaptation unit within the bounds of the master workspace to insure the tracking of the master cursor. (The actual algorithm which enforces this tracking rule, called the viewing algorithm, has not been included here.) This window relocation is

viewed at the real terminal as either vertical or horizontal scrolling.

However, an adaptation unit has the choice to bypass this rule by detaching its instance cursor from the master, effectively getting complete control of its cursor to view other portions of the master space. If the virtual display has an I/O file extension, then the adaptation unit can pan its window on the file-extended space well beyond the present contents of the master space. Therein lies the power of a stable-store data structure when coupled with the concept of windowing.

2.3 The Workstation Model

The workstation model is composed of one or more adaptation units, and a workstation monitor, which we will call the executive. Each will be described in turn below. In addition, the model includes input and output handlers, and an underlying multi-tasking operating system of unspecified architecture.

2.3.1 The Adaptation Unit

An adaptation unit embodies an instance of a virtual terminal, and since the workstation model postulates possibly many different such instances per physical workstation, then potentially many adaptation units will be co-located at a workstation.

The adaptation unit can be viewed as the workstation agent which provides the mapping between instance workspace and device workspace. To define this mapping, we introduce the notion of a viewport as a rectangular area of the physical screen allocated for the viewing of a virtual terminal instance. An adaptation unit has the task of mapping the totality of the instance workspace onto the viewport, a mapping which is a device-specific concern totally removed from the domain of discourse of the virtual terminal. Thus the position of the viewport determines the relocation of the selected data structure elements on the viewing unit, and the viewport dimensions a (potential) scaling transformation.

The adaptation unit also produces virtual input to the virtual terminal by translating the user input into virtual terminal commands. It implements the service side of the interface to the virtual terminal.

2.3.2 The Executive

This conceptual entity performs the task and resource management required to create and destroy virtual terminal instances, and to map these virtual terminal instances to the screen viewports.

It must provide at least a minimal user command interface so that its tools may be accessed (one of them being the management of screen real estate).

Finally, the executive provides the mechanism for the end-user to switch viewport contexts through the use of some input device (e.g., function key, pointing or positioning device). Following a user interaction which indicates a change of context, the executive makes the newly selected virtual terminal instance the dedicated owner of the input devices.

REFERENCES

1. R. Bisbey II and D. Hollingworth. "A distributable, display-device-independent vector graphics system for the military command and control environment," Information Sciences Institute, Marina del Rey, California, April 1978.

2. Alan Branden, et al. "Lisp Machine Project Report." Artificial Intelligence Laboratory, Massachusetts Institute of Technology, AIM 444, August 1977.

3. John Day. "TELNET Data Entry Terminal Option," ARPA Network Working Group RFC 732, Network Information Center, SRI International, September 1977.

4. Douglas Gerhart and D. L. Parnas. WINDOW A formally specified graphics based text editor, Computer Science Department, Carnegie-Mellon University, June 1973.

5. B. W. Lampson and R. F. Sproull, "An Open Operating System for a Single-User Machine," Proc 7th Symposium on Operating Systems Principles 9-17, ACM, December 1979.

6. Keith Lantz. Uniform Interfaces for Distributed Systems, Ph.D. thesis, University of Rochester, Rochester, N.Y., May 1980.

7. Mathis, J.E., et al, "Terminal Interface Unit Notebook," Volume 2, ARPA Order No. 2302, SRI Project No. 6933, SRI International, Menlo Park, California, 1979.

8. Allen Newell, Scott Fahlman, Bob Sproull. "A Proposal for Personal Scientific Computing," Department of Computer Science, Carnegie-Mellon University, July 1979 (DRAFT).

9. "PERQ," Three Rivers Computer Corp., 160 N. Craig St., Pittsburgh, Pa. 15213.

10. Jon Postel and Dave Crocker. "TELNET Remote Controlled Transmission and Echoing Option," ARPA Network Working Group RFC 726, Network Information Center, SRI International, March 1977.

11. John F. Shoch and Jon A. Hupp. "Notes on the 'Worm' programs -- some early experience with a distributed computation," Xerox Palo Alto Research Center publication SSL-80-3. Presented at the Workshop on Fundamental Issues in Distributed Computing, ACM/SIGOPS and ACM/SIGPLAN, December 1980.

12. R. F. Sproull and E. L. Thomas. A network graphics protocol, Computer Graphics 8(3), Fall 1974.

13. C. P. Thacker, E. M. McCreight, B. W. Lampson, R. F. Sproull, and D. R. Boggs. "Alto: A Personal Computer." D. Siewiorek, C. G. Bell, and A. Newell, Computer Structures Readings and Examples, editors, second edition, McGraw-Hill, 1979.